N91-22789

A Reusable Knowledge Acquisition Shell - KASH

Christopher Westphal
Stephen Williams
Virginia Keech

SYSCON Corporation
1000 Thomas Jefferson St., N.W.
Washington, D.C. 20007

Abstract

KASH (Knowledge Acquisition SHell) is proposed to assist a knowledge engineer by providing a set of utilities for constructing knowledge acquisition sessions based on interviewing techniques. The information elicited from domain experts during the sessions will be guided by a question dependency graph (QDG). The QDG, defined by the knowledge engineer, will consist of a series of control questions about the domain that are used to organize the knowledge of an expert. The content information supplied by the expert, in response to the questions, will be represented in the form of a concept map. These maps can be constructed in a top-down or bottom-up manner by the QDG and used by KASH to generate the rules for a large class of expert system domains. Additionally, the concept maps can support the representation of temporal knowledge. The high degree of reusability encountered in the QDG and concept maps in KASH can vastly reduce the development times and costs associated with producing intelligent decision aids, training programs, and process control functions.

Introduction

The field of expert systems has claimed many successful applications ranging from simple tasks, such as monitoring a valve or fluid rate, to very complex tasks that may include process scheduling or design. However, to construct a system requires the developer to isolate a set of rules that will guide the decision making process. Rules are conventionally defined during a series of interviews between a domain expert and a knowledge engineer. The encoding and representation of the extracted domain knowledge have proven to be difficult barriers to overcome and have been commonly deemed the *knowledge acquisition bottleneck*. To reduce the amount of time required for this stage of expert systems development, knowledge acquisition techniques have been extended beyond the traditional verbal interview methods to include both semi-automated tools that improve knowledge engineering efficiency and fully autonomous approaches (machine learning) that attempt to infer knowledge directly from examples of expert behavior.

To date, numerous tools have been developed to address the many aspects of knowledge acquisition. The tools tend to be highly user oriented with the interfaces and structure of information significantly well advanced to accommodate a rapid and easy facilitation of knowledge. The application domains of these tools can be aligned into two categories; *analysis* (e.g., diagnosis, identification, interpretation) and *synthesis* (e.g., scheduling, planning, design). Analysis is a top-down process providing an examination of a set of goals that are decomposed into simpler and more basic elements and relationships. For example, FIX [Rodi, Pierce and Dalton, 1989] is a diagnostic system based on class attributes for representing fault detection and isolation. Synthesis is a bottom-up process supporting the composition or combination of given facts and observations that are used to construct a set of goals. For example, SALT [Marcus, 1988] is a tool based on a propose-and-revise method for designing elevator configurations.

The above classes of tools work well within their intended domains, but are not easily transferred to other applications. Consequently, the time and effort required for producing the systems must be reinvested for each instance. This will increase the costs to the developers and subsequently decrease reusability and

portability of the tools across domain applications. To address the different categories (analysis or synthesis), a system must be able to work within the constraints set by each. Expert system shells (e.g., KEE, ART, CLIPS, NEXPERT) currently provide the capacity to build top-down or bottom-up applications by supplying the developer with a finite collection of objects to represent the domains. The manner in which the objects are defined and related to other objects determines the degree of acceptability of the system. Therefore, if the focus of producing a knowledge acquisition system is placed on the *structure* rather than on the *content* of knowledge, a more diverse selection of applications may be addressed.

Yet, much of the previous research with knowledge acquisition tools has been confined to specialized areas of interest with minimal attention allotted to the dissemination of the information. Using the control knowledge represented in the structure of information will facilitate the reuse of knowledge between applications, and the context information can then be elicited for the structures from the experts for each unique application. Slight variances in the control structures can radically affect the content information obtained. Multiple expert based systems are naturally one of the primary benefits of this type of approach. Thus, the domains of interest can extend across a large number of applications including training systems, mission planning, and manufacturing. In doing so, the standardization of knowledge and the issues of reusability become a reality.

Reusing Knowledge Structures

The reusability issues of knowledge based systems have slowly moved into mainstream considerations of expert system developers. Many government, industry, and academic sectors have participated in the development of expert systems technologies and have produced a considerable set of disheveled applications. The information obtained from the knowledge engineering phase, much of which is repeated between applications, is typically the largest cost of expert systems production [McGraw, 1989]. If a reusable data store of reliable information could be produced, the development costs for each application would be reduced significantly. Furthermore, it would provide a basis from which to partially (or wholly) support *standardized* knowledge based representations.

The introduction of reusable knowledge components is a convenient mechanism for bridging domain applications. An instrument capable of accommodating the various knowledge structures encountered in any application would require a representation that focuses on the structure of information. The content of these structures could then be filled in later by the appropriate domain experts. Surprisingly, very few knowledge acquisition tools have been designed with this type of approach and even fewer to address multiple application domains.

One knowledge acquisition system providing a domain-specific structure, KLAMShell [Cochran, 1988], was developed to aid in the construction of knowledge bases for maintenance and troubleshooting. The shell elicits information via subgoal satisfaction in a depth first manner to retain a context focus on the knowledge structure. Each goal defined is decomposed (via "push" menus) into a series of subgoals. The bottom-most nodes are then transformed (via "pop" menus) into actions, such as questions or instructions, to be used in the final system. This process continues until all goals have been specified. The system is, however, domain specific and the generality of its guidance is limited to only a *small* subset of the domain, thus restricting its use. Another system that may be classified as a general purpose knowledge acquisition shell is PROTEGE [Musen, 1989]. It is a tool capable of generating other knowledge acquisition systems using planning entities, task level actions, and input data. The methodologies used by PROTEGE separate the modeling of a domain (i.e., control structure) from the application knowledge, thereby customizing each system. However PROTEGE relies on a set of fixed templates to elicit information and this will limit the range of systems that can be produced.

From this perspective, KASH (Knowledge Acquisition SHell) has been proposed as a domain-independent knowledge acquisition shell. The shell will provide a general purpose (reusable) environment for encoding the problem-solving methods of domain experts. A set of three independent modules (Figure 1) have been defined in KASH that will operate within the top-down (analysis) and bottom-up (synthesis) constraints of various applications. These modules are: *Concept Formulation*, for eliciting knowledge from domain experts and structuring it into concept maps; *Knowledge Analysis*, to verify the concept maps and cross check any inconsistencies found; and *Rule Generation*, to produce the rule sets for the target expert system shells based on the concept maps. Temporal knowledge will be generated by the rule structures and stored in the concept maps. A time-box will supply a graphical view of the time intervals specified and a time-line analysis will show the instances of time for a particular execution of the system. The question dependency graph (QDG), which is responsible for guiding the interview sessions, will supply the control structures used in the concept formulation module. Control structures, defined *a priori* by knowledge engineers, are necessary to obtain the content knowledge from the experts. The utility of the QDG allows the control structures to be generated as needed and variations of the graph can easily address new applications.

Organizing Knowledge in KASH

The basis for acquiring knowledge in KASH is by organizing the cognitive processes an expert formulates about a particular domain. The mechanisms to organize the processes must be tailored to the idiosyncratic representations created by

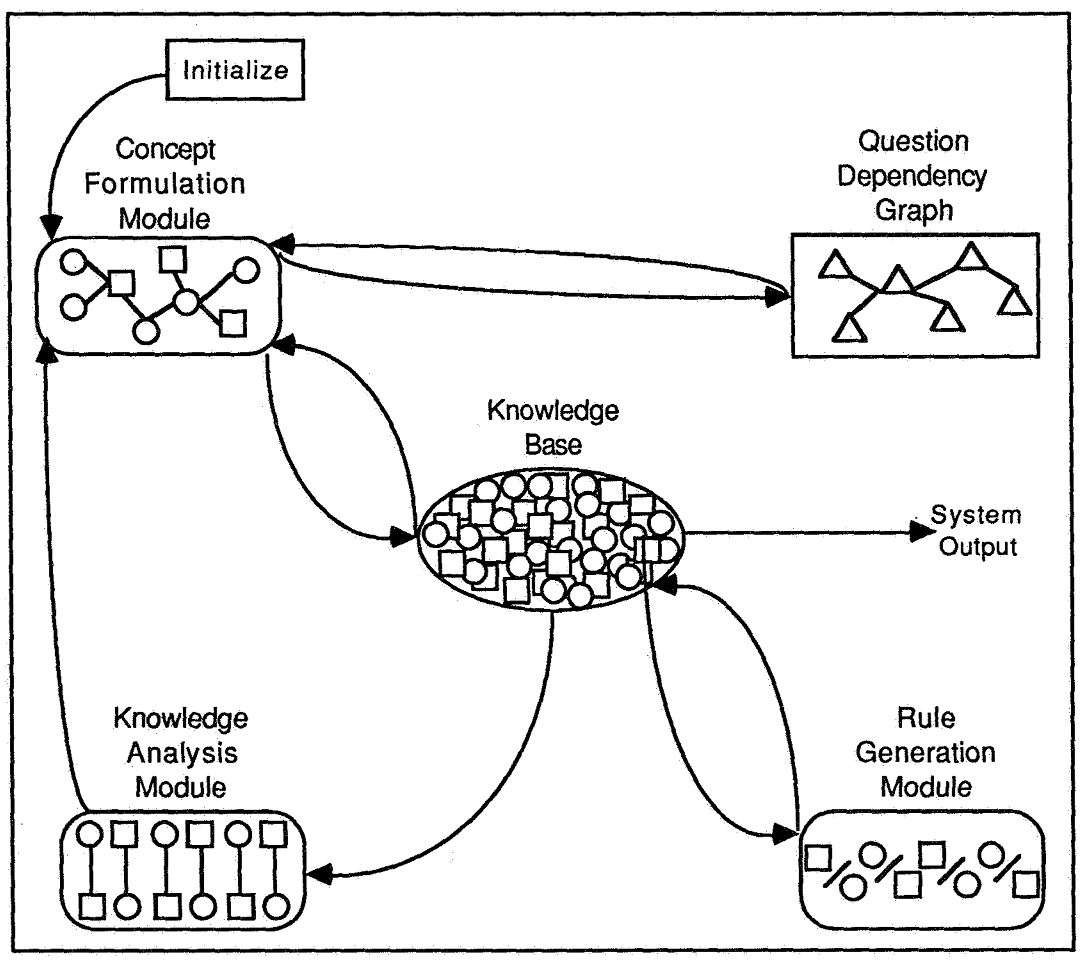

Figure 1. KASH Architecture

the experts. However, experts tend to express their knowledge in unstructured and nondeterministic formats and difficulties arise when the knowledge must be converted into a reliable and useable format for expert systems development. To properly address these representation problems, the experts must be allowed, with minimal constraints, to describe their cognitive processes with respect to the intended application. Typically, the experts will define concepts that represent personal observations in the domain to be modeled. Concepts are defined as symbols (text or image) that capture the meaning or intention of objects and events of an environment [Ausubel, Novak and Hanesian, 1978]. For example, an "electrical system" is an object, and "charge battery" is an event. In essence, a concept is a *simplified* and *generalized* description of reality for a particular level of abstraction [Novak and Gowin, 1984].

Concept formulation may be defined as the process of extracting the characteristic features, termed criterial attributes, of the objects or events. The criterial attributes are what uniquely distinguish each concept (e.g., size, color, shape). A set of common features and the degree to which the features are accepted by the experts will determine the regularity (i.e., meaning) of a concept. The regularity implies precision on the definition of the concept so that misunderstandings can be minimized, see [Novak and Gowin, 1984]. Criterial attributes also allow concepts to be grouped or linked together to form concept maps that can represent the conceptual and structural knowledge of the domain experts. A concept map is simply a hierarchical taxonomy of concepts. The hierarchy supports the natural subsumption of concepts where broad concept definitions are depicted at the top of the map, and more detailed and concise concepts at the bottom. The concept hierarchy is analogous to the familiar object hierarchy successfully applied in expert systems development. Concept maps promote the reuse of concept definitions by allowing the use of existing concepts (e.g., door) in different contexts (e.g., house, car). Such a virtual feature is natural for multiexpert integration because it provides support for a modular structure and the development of a library facility of reusable concept definitions across domains where applicable [McGraw and Westphal, 1988]. The graphical nature of concept maps also allows multiexpert conflict to be made explicit, thereby excising the knowledge structure of faulty linkages and misconceptions [Westphal and Reeker, 1990].

The concept map in KASH has been extended to include base facts [Westphal and Tran, 1990]. Base facts have their own definition because they are fully instantiated (e.g., parts, ingredients, symptoms) and do not require further justification for their existence. Base facts, therefore, provide logical grounding for reasoning performed over the concept hierarchy. Base facts are measurable and observable in the context of the concept, and support the criterial attributes of a domain. The criterial attributes are necessary for distinguishing between concepts and structuring questions into categories during concept formulation. The criterial attributes also facilitate the rule generation module because they support variable entities (e.g., size is big, color is red) that can be compared, contrasted, or combined with other criterial attributes to form the rules produced by KASH.

Question Dependency Graph

The development of a concept map requires that the experts be interviewed using questions pertinent to the content and structure of the problem domain. As [Novak, 1989] stipulates, the sequencing of questions presented to the expert should be tailored or grouped into specific sets of knowledge, and the range of these sets should address questions at broad superordinate levels and become more narrow and precisely defined at the base level. In KASH, the questions permissible to ask of the domain experts during the development of the concept map will be specified by the knowledge engineers and represented in a question dependency graph (QDG). The QDG is abstractly based on the six types of questions as defined by LaFrance [1987] where each question type will decompose the QDG into a category of

queries (broad and specific) to be asked with respect to conceptual definition. The six types defined are:

- *Grand Tour*, to identify domain and subdomain boundaries, e.g., "What is the purpose of this system?"
- *Cataloging the Categories*, to support, organize, and structure the concepts, e.g., "Can the concepts be ordered?"
- *Ascertaining the Attributes*, to specify values and ranges of the criterial attributes for concepts and base facts, e.g., "What values can attribute assume?"
- *Determining the Interconnects*, to define the causal relationships between concepts, e.g., "Does base fact support the concept?"
- *Seeking Advice*, to obtain expert recommendations and determine special conditions e.g., "Does concept contain any base facts?"
- *Cross Checking*, to compare and contrast information in the concept map, e.g., "What value of attribute is out of range?"

During the development of a QDG, a knowledge engineer will specify the questions to be asked of the domain experts. The questions will be formulated to separate the control knowledge from content knowledge of the domain (analysis or synthesis). Thus, the QDG is a form of meta-knowledge that specifies the information about how to elicit domain-level knowledge from the experts. The meta-level knowledge of a QDG implicitly tends to be high-level in nature and can be constructed as follows:

- The first version of a QDG will be obtained from a library of QDG bases. The libraries supplied to the knowledge engineers will be provided with the initial version of KASH. The QDGs will be the result of a study of the elicitation process for a particular domain (analysis or synthesis) and will cover the basic acquisition of rule constructs, temporal intervals, and a variety of generic representation mechanisms.

- The knowledge engineers will further refine and specialize the existing QDGs to meet the requirements of their application. Therefore it will be important for the knowledge engineer to understand the basic structure of the domain to effectively define questions, categorize the questions by type, and determine the control sequence of the questions. Information can be obtained through a domain analysis of technical literature, existing implementations, surveys, and system requirements. Inefficient specifications at this phase can lead to poor interview sessions with the domain expert, who will supply the content knowledge (responses to the questions).

In the QDG, the questions are expected to be represented as node structures and will be linked to other node structures to determine the sequencing. The node structure design will consist of a question frame, question type, selection guide, and a variety of support attributes. The question frame will contain the actual text generated during a query. Questions are developed by the knowledge engineers about the control structure of the domain through textbook information, job analyses, and previous case studies. The questions derived will be used to respectively refine or compose the concepts in an analysis or synthesis domain and will be continually refined through interaction among knowledge engineers and domain experts. The six question types previously defined will be used to ensure the scope of the questions are limited to the current query category. By doing so, the focus of the experts will be on the particular task level being modeled. The selection guide will be satisfied when the knowledge engineer specifies the response expected from the question frame with respect to one (or several) of the system objects specified in KASH. These objects are expected to be a prioritized taxonomy of concepts, base facts, edges, attributes, and standard representation mechanisms (i.e., integers, reals, text) that will help resolve which question to ask.

The edges will form the explicit relationships between the node structures and determine which tuples of questions may be presented to the domain experts. A tuple of two questions is interpreted to imply: if question one is asked, then question two may be asked. There will be many-to-many relationships between the node structures, thus forming a large combination of questions tuples. The edges will be coupled with edge weights to determine the ordering of question preference, a form of prefiltering. These weights will be displayed as a range of numbers or as an alternative representation.

concept, and a miscellaneous collection of support attributes helps to specify display formats (DF) and the control strategy (CS). The links between the nodes are labeled with high, medium, and low priorities to indicate the control paths the system should pursue. In this example node D is connected to nodes E and F. The heuristics defined in the concept formulation module will select the better candidate question (E or F) to ask based on the state of the concept map.

A QDG editor will be used for the construction of the question graphs. The editor will allow the knowledge engineers

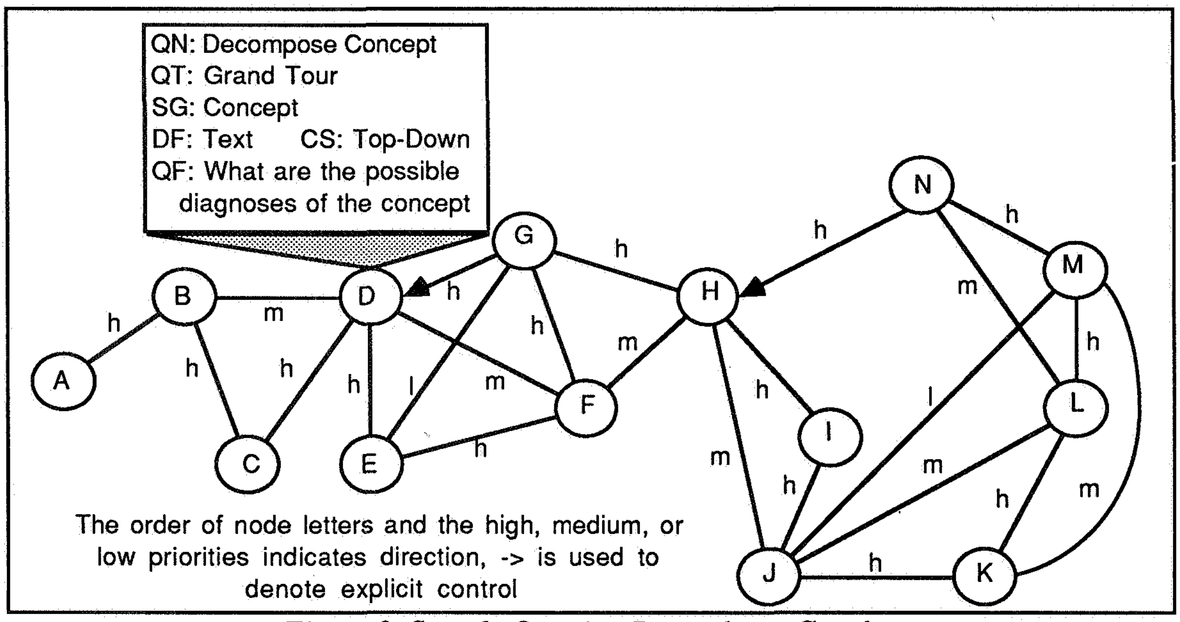

Figure 2. Sample Question Dependency Graph

Figure 2 shows an example of a partial question dependency graph for a circuit fault diagnosis. Each circle represents a unique question (defined by a knowledge engineer) that may be asked during a knowledge acquisition session. The box represents the node structure (only one is detailed) for a particular query scheme. The question name (QN) is Decompose Concept, the question type (QT) is Grand Tour, the question frame (QF) is defined as "What are the possible diagnoses of the concept?" (variable terms in the question will get instantiated during execution), the selection guide (SG) is expecting the response of the domain expert to be a

to construct specialized question bases through QDG link specifications. New questions may be introduced into the graph at any time. The set of base questions will always be available, however it is the format in which they are connected that will give a system its functionality. This combination of nodes and links will allow KASH to be used across a large range of applications and will prove to be very versatile and powerful throughout the knowledge modeling process. As the QDG changes, so will the system, because the QDG is the canonical guidance referenced throughout the three modules of KASH.

As previously mentioned, the KASH system has been designed as a set of three (3) independent modules to support a reusable knowledge environment. The modules are *Concept Formulation*, to obtain and structure knowledge from an expert; *Knowledge Analysis*, to validate the knowledge structures; and *Rule Generation*, to produce a set of rules based on the acquired knowledge. Temporal knowledge is addressed explicitly by KASH through the rule generation module. Each module will be discussed in detail below. Additionally, a set of support modules will also be defined to enhance the user interface.

Concept Formulation

Concept formulation, necessary to acquire the knowledge structure from a domain expert, is interleaved between a top-down and bottom-up elicitation strategy guided by the QDG. The top-down refinements are focused on acquiring the abstract concepts that are used to define the components of a domain. These components are in turn supported by bottom-up facts that represent the logical entities or extension of the concepts. The use of top-down and bottom-up strategies in concept formulation reflect the structures that exist in the different levels of knowledge and the control nature of the analysis and synthesis classes of problems. Ausubel [1978] addresses these abstractions in his propositional learning theory. The analysis problems involve identifying sets of objects based on their features. Synthesis problems construct a solution from component pieces or subproblem solutions. Figure 3 is a section of the top-down (and bottom-up) process trace of a concept formulation based on the QDG for a circuit fault diagnosis. The index (A-N) for each instantiated question text shown (i.e., variables are bound within the context of the question) corresponds to a node in the QDG shown in Figure 2.

The concept formulation module will iterate through six stages during the construction of a concept map. These stages are responsible for selecting a concept from the graph, establishing base facts for the concept, formulating a question to apply to the concept, generating the question, then presenting and accepting the results from the expert. The six stages are described below and their logic flow is displayed in Figure 4.

A)Define the terminology:
Top-down terms(s) :> diagnosis.
Bottom-up terms(s) :> symptoms.

B)What is the purpose of this diagnosis?
:>circuit fault.

C)Define any symptoms of a circuit fault.
:>manifold threads.

D)What are the possible diagnoses of a circuit fault?
:> control circuit failure.
:> sensor fault.

E)Can these be ordered?
:> no.

F)Is manifold threads a symptom of control circuit failure, sensor fault?
:> sensor fault.

G)Are there any other symptoms of sensor fault?
:> sensor mounts.

D)What are the possible diagnoses of sensor fault?
:> broken sensor.
:> sensor grounding.

E)Can these be ordered?
:> no.

F)Is manifold threads a symptom of sensor grounding, broken sensor?
:> sensor grounding.

F)Is sensor mounts a symptom of sensor grounding, broken sensor?
:> sensor grounding.

G)Are there any other symptoms of sensor grounding?
:> voltage test.

D)What are the possible diagnoses of sensor grounding?
:> none.

D)What are the possible diagnoses of broken sensor?
:> none.

H)What are some attributes of sensor mounts?
:> connectors.

I)What value(s) can connectors of sensor mounts assume?
:> loose, broken, grounded, working.

J)Is connectors used in sensor grounding?
:> yes

K)What value(s) of connectors would make sensor grounding succeed?
:> grounded, working.

L)What value(s) of connectors would make sensor grounding fail?
:> loose, broken.

M)Is this a default or inferred value?
:> inferred.

N)Can connectors be related to other attributes?
:> no.

Figure 3. Dialog Trace of Sample Interview Session

Heuristic Criteria for Concept Selection - A concept will be chosen from the set of open concepts. This set represents those concepts that have not been fully explored by the QDG such that there are additional questions that may be asked about them. The concept selection will be based on several factors, including the order of importance that is explicitly specified by the user, a system defined depth-first or breadth-first selection mode, the recency of a concept definition, the level of attributes defined, the number of established links to base facts, the number of derived (children) concepts, or a concept of enumerated type.

Establish Base Facts - The first step taken after the concept has been selected will be to elicit the base facts that support the concept. The base facts define the fundamental units of the domain. New base facts will be specified by the user and the supportive links established to the selected concept. User selected base facts will be passed down to the selected concept from a parent concept. Note, parent concepts must resolve all supportive links before the concept map is considered complete.

Select Questions - The list of applicable questions for the current concept will be calculated by obtaining all the edges attached to the previous question in the QDG and placing them into a list. A duplication filter will be applied to the list to resolve any conflicts found in a list of previously asked questions maintained as a catenation of the concept name and the question identifier. The list can then be processed through some ordering filters that act on the edge weights, selection guide matching, and group division types. The product of these filters will

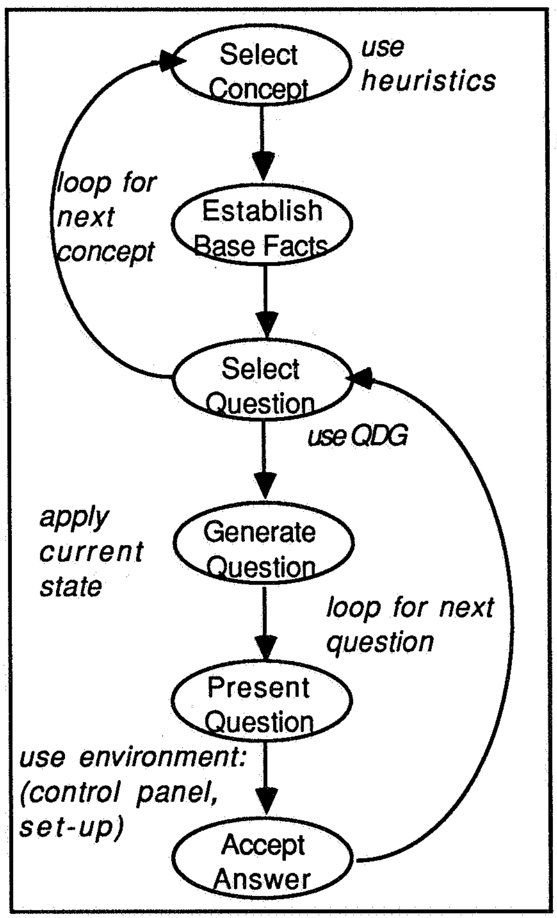

Figure 4.

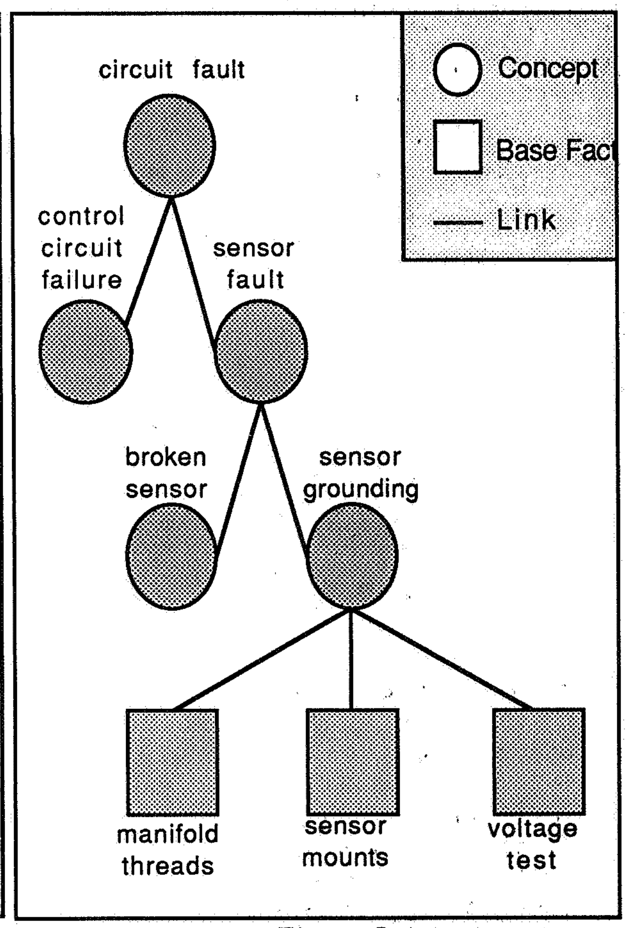

Figure 5.

result in a refined list of questions to be asked.

Generate Question - The product of the generate question stage will be the selection of a single query node made from the refined list. This list of applicable questions for the current concept will be acted on by a function of the question types and other system factors. A priority scale will be produced and the query with the highest rating will be selected.

Question Presentation - After a single question has been selected it will be presented to the user to solicit a response. The control panel will be consulted for the appropriate terms (top down/bottom up), display formats (bar graph, text, pie chart, menu), and concept graph information (attributes, types, links).

Answer Questions - The expert will respond to the question through either text (e.g., attributes, concepts, base facts, names, or values) or concept graph management facilities (e.g., graphics or node selection).

The output expected from the concept formulation module will be a concept map representing either a top-down, bottom-up, or combination process model. The concept map will represent the observations of domain experts as guided by the control structure specified in the QDG. Figure 5 shows the concept map for the circuit fault example. If the information acquired does not fully define the range of an expert's knowledge, the QDG may be reconfigured to accommodate the missing components. From this point the information gleaned from the expert in the form of concept modules and base facts can be verified by both the system and the domain experts. Several structure analysis techniques, described below, have been defined to accommodate this task.

Knowledge Analysis

The knowledge analysis module will be a hybrid of several interactive subsystems that are used to validate and refine the concept map. During validation, the knowledge acquired from the concept formulation module will be analyzed to bring out inconsistent, incomplete, and unjustified information that may occur in particularly large systems or from the use of multiple experts. In refining the concept map, the system will look for ways to enhance the structure of the model through the application of a variety of techniques. Several proposed techniques are described below and in the accompanying Figure 6 (a-e).

Cluster Analysis - Cluster analysis will detect cases in which large numbers of concepts are derived from (or clustered around) a more general concept. It suggests that the general concept is too broad, and it should imply several new concepts where the other concepts can be derived *from*, or some derived concepts should be promoted *to* the intermediate level concepts between the general concept and others. The introduction of intermediate concepts helps structure the knowledge elicitation stage and reduces the complexity of the acquired knowledge. Figure 6a.

Entailment Analysis - The basis for entailment analysis stems from the attributes associated with the concepts. The technique will make the entailments between concepts obvious and will indicate that a concept can be subsumed by another concept because it is derived by another concept that has no other derivation. Figure 6b.

Relative Analysis - Relative analysis is based on the reuse of concept modules in the graph. A concept associated with a base fact will be selected for review. The analysis technique will propagate up the structure to a level outside the immediate hierarchy associated with the concept, and query the expert, via the QDG, if it can be used in the adjacent paths. This type of analysis will support the use of virtual structures and serves as a form of memory cue entailment for additional structure refinements. Figure 6c.

Subcomponent Analysis - A base fact will be selected from an active path in the concept structure. The path will

consist of the base fact and all the concepts that comprise the path to the root node. The base fact will be compared to each of the concepts that subscribe to the path and questioned by the QDG for relevancy against the concept. Negative responses will indicate inconsistencies in the structure and will be resolved through further QDG examinations. Figure 6d.

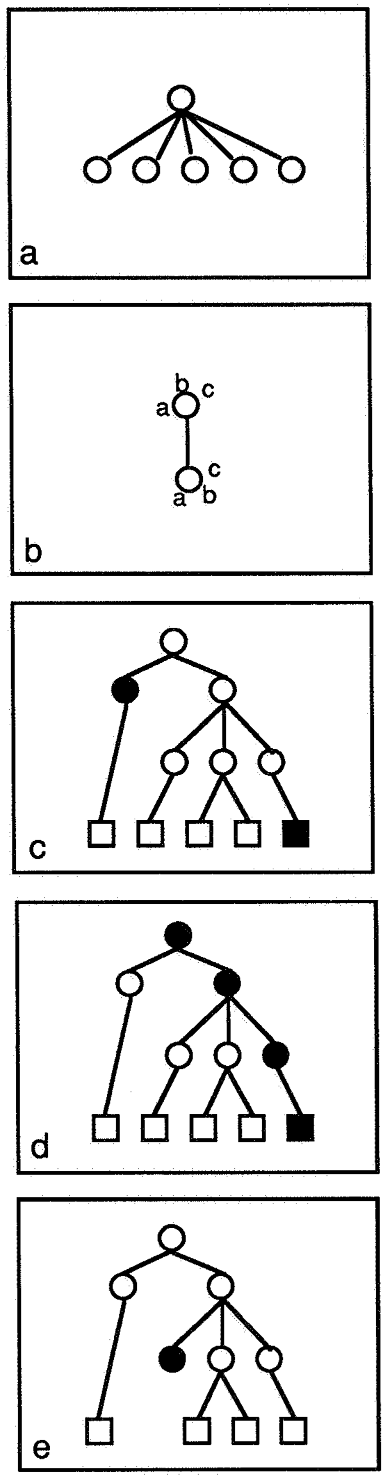

Figure 6 (a-e).

Conceptual Analysis - This attempts to enforce the principle that every base fact must be attached to a concept that is not further decomposed by additional concepts. If any such case is encountered it can be assumed that there will be additional conceptual levels that have not been defined. Additionally, if a concept has been decomposed into another single concept, then the linear formation will invoke the analysis to collapse both into a single concept structure. Figure 6e.

Knowledge analysis will be interactive or selectable. In the first mode, *interactive*, a number of knowledge analysis techniques will be selected from a menu before the concept formulation module is executed. As knowledge is acquired the techniques will be automatically invoked to yield analysis results. The results will be shown in a side window with the appropriate marks to indicate the different degrees of importance. In the second mode, *selection*, the knowledge analysis stage will be entered after a knowledge elicitation session. The manner in which the analysis results are shown will be the same as in the previous mode, but more thorough analyses can be applied.

Rule Generation

The third module of KASH will be used to support the final process of knowledge acquisition, rule generation. This module will create a set of rules that can be integrated into a third party expert system shell. The three major factors that will influence the generation of the rules are the concept map, rule generation strategy, and target expert system shell.

First, the concept map will be extended to include any terms (criterial attributes) necessary for the production of the rules. New attributes will be created and manipulated by *attribute<->attribute*, *attribute<->concept*, and *attribute<->value* questioning schemes defined in the QDG. The information provided by the experts in

response to the questions will specify the attributes of a concept, the relationships of the attributes to other attributes, and the range of values the attributes may assume. Attribute definitions will be initiated for each base fact in the concept map. The definitions will be propagated to the concepts via an inverted inheritance scheme with question filters to terminate any attribute stream. The attribute relationships in the concepts will represent rules. The rules will be local to the concept for which they are defined and may be classified as either *default*, where new logical relationships between attributes will be specified (e.g., set the temperature of material to equal the temperature of the gas), or *inferred* where the system must calculate the value of the used attribute (e.g., does the temperature of the material equal the temperature of the gas). The inferred values will be present in the rule premises and the default values will typically appear in the consequents.

Second, in the rule generation strategy, the local rules will be used by a series of high level top-down and bottom-up strategies that determine the direction in which final rules are to be developed. The top-down strategy employed in generating rules will be based on a concept and the links to its supportive concepts. The bottom-up strategy will direct the rule generation based on the lower level concepts to the higher level concepts that they support. This design will provide both forward and backward chaining rule specifications for integration into the expert system.

Third, the characteristic or model of the target expert system shell language will influence the different ways in which an optimal rule set may be generated. This will depend on the different representation schemes, control strategies, daemons, triggers, etc. encountered in the different expert system shells. For example, instead of explicitly stating separate rules as would be necessary in a pure production system such as OPS-5, the number of rules generated can be reduced in an object oriented system such as NEXPERT, CLIPS, or KEE due to the abstraction of frame structures.

```
Rule 1                                              Rule 2
IF level_of_voltage_test < 3                        IF level_of_voltage_test >= 3
THEN position_of_sensor_grounding = "open"          AND level_of_voltage_test < 5
                                                    THEN position_of_sensor_grounding = "closed"

Rule 3                                              Rule 4
IF connectors_of_sensor_mounts = "loose"            IF connectors_of_sensor_mounts = "broken"
THEN cond_of_sensor_grounding = "poor"              THEN cond_of_sensor_grounding = "poor"
AND recd_of_sensor_grounding = "tighten"            AND recd_of_sensor_grounding = "replace"

Rule 5                                              Rule 6
IF connectors_of_manifold_threads = "dirty"         IF connectors_of_manifold_threads = "corroded"
THEN cond_of_sensor_grounding = "poor"              THEN cond_of_sensor_grounding = "poor"
AND recd_of_sensor_grounding = "clean"              AND recd_of_sensor_grounding = "replace"

Rule 7                                              Rule 8
IF cond_of_sensor_grounding = "poor"                IF cond_of_sensor_grounding = "poor"
AND position_of_sensor_grounding = "open"           AND position_of_sensor_grounding = "closed"
THEN status_of_sensor_fault = "true"                THEN status_of_sensor_fault = "true"
AND action_of_sensor_fault = "shutdown"             AND action_of_sensor_fault = "maintenance"
AND recd_of_sensor_grounding =                      AND recd_of_sensor_grounding =
      recd_of_sensor_fault                                recd_of_sensor_fault
```

Figure 7. Partial Rule Constructs Generated by KASH

Figure 7 shows the partial set of rules that have been created for the circuit fault diagnosis example. Before these rules are converted into a target expert system format they must be reviewed by a series of analysis techniques to ensure there are no redundant, conflicting, or circular rules.

The outcome of the rule generation system module will be a knowledge base of objects, relationships, and rules that could support a variety of expert system applications.

Representation

The underlying information detailed during the construction of the concept map, QDG, and rules are expected to be stored in a frame-structure representation. The frame-structure, originally proposed in the 1970s, supports the modularity necessary to easily add, modify, or delete information from the knowledge structure. It is the degree of modularity encountered in a system that directly promotes the decomposability of an application [Simon, 1969], as may be seen during the construction of the concept map. The KASH architecture has been designed with minimal dependencies between its representations such that all question nodes in the QDG and sections of the concept map can be extracted, interpreted, and reused in other applications. Furthermore, the slot values associated with frame-structures will provide KASH with one mechanism to confirm a complete specification of the rule constructs.

Temporal Knowledge

Representing and reasoning over time is an important factor to consider when developing expert systems, especially in domains such as evaluation, planning, and scheduling. Although there has been a large effort of work associated with time, very little research has been conducted with respect to temporal knowledge acquisition. The KASH system will address this unfledged technology by eliciting *numerical* and *symbolic* time references through the use of a time-box facility and time-line analysis. It is expected that temporal knowledge will be elicited (when required) through an extension of the rule-generation module.

The hierarchical structure of concept maps forms a type of temporal circumscription where absolute time intervals (*numerical*) can be applied without ambiguity in meaning. The specification of these intervals is straightforward and done by the domain expert for all relevant concepts encountered in the graph by limiting the duration specified for a concept to its lineal descendants. When a time interval has been stated for a primary concept, any subsequent intervals stated for its dependent concepts must not exceed the initial interval; the duration of the parts must be equal to or less than the whole. For example, if concept A is supported by concepts B and C and the duration set for A is <u>five</u> minutes, then the total time to execute B and C should not exceed this interval. Start, end, and latency times are required for numerical intervals. The decomposable nature of concept maps supports temporal relations (*symbolic*). Concepts are composed *of* or derived *from* other low or high level descriptions (concepts or base facts) of the application. Thus, when describing the temporal relationship between two concepts, Allen's [1983] enumerated set of time primitives may be used. For example, concept B must occur <u>during</u> concept A. Utilization of these primitives also provides a set-theoretic method of reasoning about temporal relationships. Both the absolute and relative time constructs will be easily accommodated by the slots associated with the frame representation used by KASH.

The questions concerning the representation of time constructs will be defined in the QDG jointly under the determining-the-interconnections and ascertaining-the-attributes classifications. Examples of such questions will include "Can the concepts be ordered?" or, "Does concept_x occur before concept_y?" or, "What is the start time of concept_z?" A time-box has been defined (Figure 8) that will depict the results of the temporal specifications for the concepts in a graphical manner. The horizontal axis of the time-box represents the forward notion of time and the vertical axis is used to display the depth or level of the concepts. Recurring patterns will be represented in the time-box as open-ended (broken) time slices and are defined as those instances where the activities occurring during a particular series of time units are repeated more than once (see intervals E, L, & O). In KASH the repeating section will be related to a triggering event (i.e., a specific attribute

value in a rule sequence) in the superordinate time interval. For every instance the event is encountered, the corresponding activities will repeat for their defined intervals. All subsequent time executions will be adjusted for multiple successions of a repeating event, thus maintaining a dynamic consistency across the temporal span. The period of 'Mondays' in a particular month is an example of a recurring pattern as defined by [Ladkin, 1988] and would be composed as the union of each unique 'Monday' for that month interval. The triggering event here may be the requirement of a staff meeting every Monday morning.

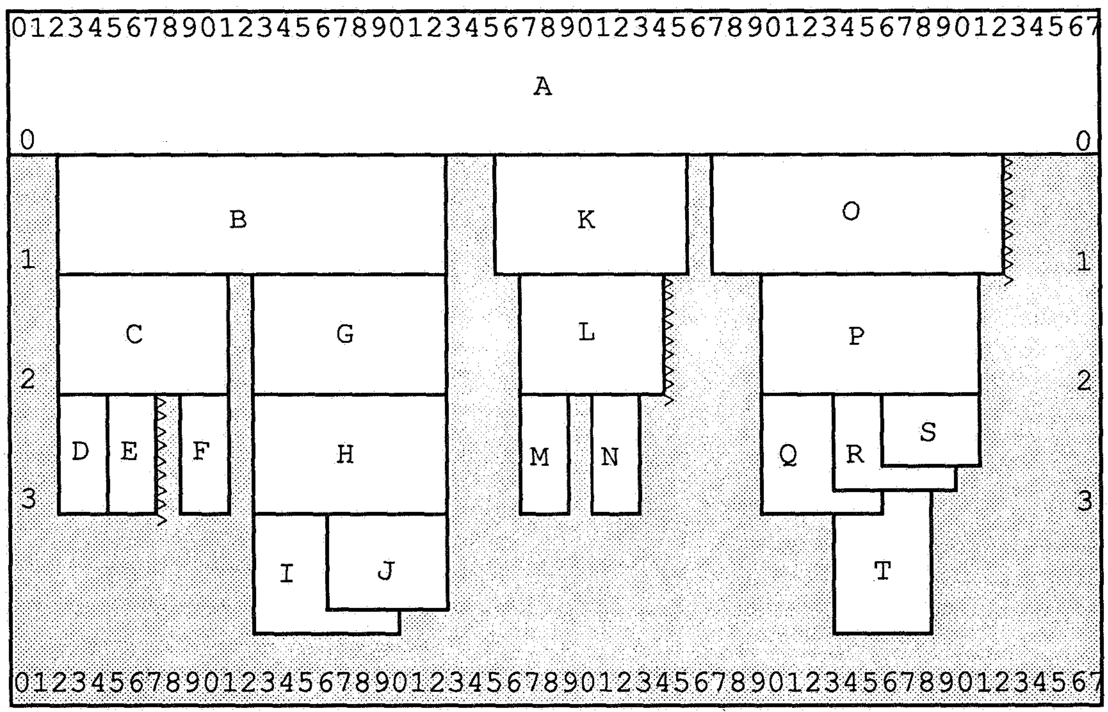

Figure 8. Time Box

A time-line analysis complements the time box. Whereas the time-box will be used to define time for the structure of the application, the time-line will show the actual instances (persistence) of time for an execution of the system, see Figure 9. A unit of time will represent an interval where the state of the system will be allowed to change. Many instantaneous changes can occur within a time unit, but it is the final values that will be recorded and stored in the time-line analysis. For example the rule "If the temperature_of_water > 212 for more_than 3_minutes Then set status_of_water = boil" would examine the current state of the time-line for the slot *temperature* of the concept *water* and determine if the value has been greater than *212* (unit=degree) for a duration of time exceeding three minutes. If the time-line analysis supported the antecedent condition, the slot *status* of *water* would be set to equal the value of *boil*. The time-line analysis in KASH will prove similar in functionality to that of the temporal network in ONCOCIN [Kahn, 1985] and the time mapper in KAT [Geesey, 1988]. As is expected with all results generated in KASH, the temporal information will have a series of analysis techniques that are expected to detect conflicts and unreachable situations caused by misrepresented time intervals.

Supportive Modules

There exist several external features that will be added to the basic functionality of KASH. These features are intended to extend support to the knowledge acquisition process and supplement the work of the QDG. The following represent a subset of the planned features:

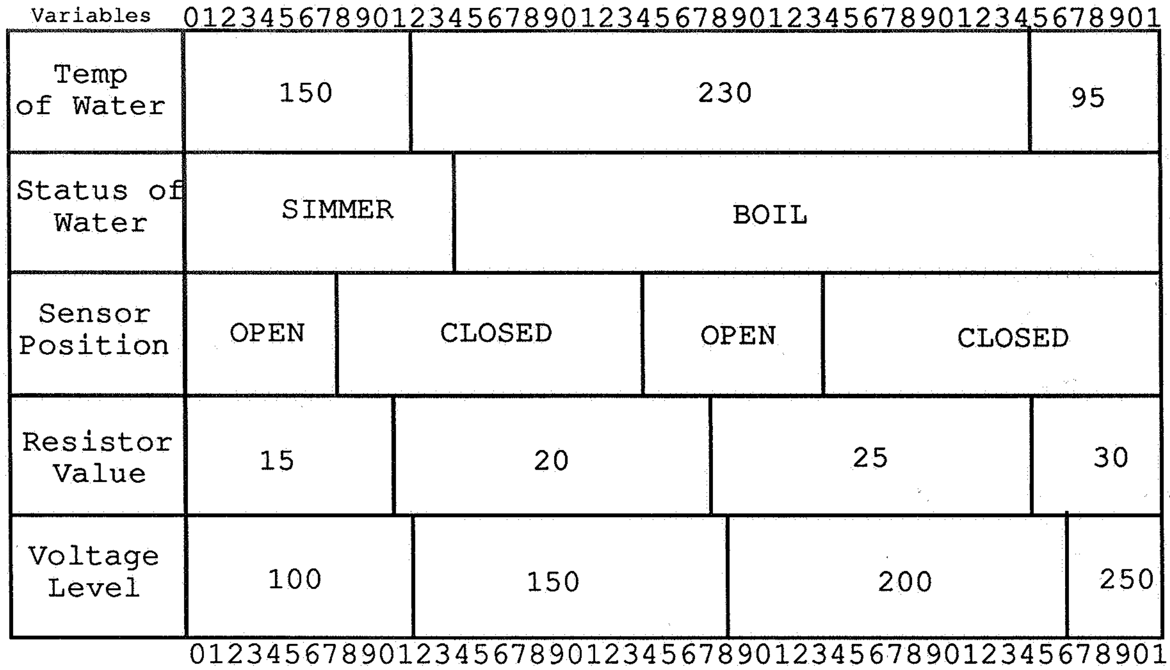

Figure 9. Time Line Analysis

• An enhanced control panel will contain a multitude of user-defined parameters to coordinate the elicitation sessions. The panel will contain the terms used for the top-down and bottom-up definitions. Pre-logical settings for which top-down entries are best suited for the bottom-up selections will be available. The search strategies (i.e., depth-first, breadth-first) can be set from the panel. A question list will be defined in the panel and used to disable groups of questions for the session (usually low priority settings). Additionally, the analysis techniques will be made interactive or selectable from the panel settings.

• A history window will supply a meta-command interpretation of the events occurring in the system. This will provide documentational support to analyze the elicitation session as it progresses over time. The log will be used to archive the state of the concept model for easy reconstruction at a future date. Additionally the history list will be constructed for path resolution of the virtual system hierarchy.

• A library window will provide interactive graphic access to previously defined concept hierarchies and QDGs. This facility will enable partial or complete merging of libraries and the current environment, thereby supporting the reuse of KASH structures.

• A glossary of terms defined by the expert during the elicitation of concepts and base facts will be supported as a scrolling series of menus. This facility can support the comments associated with multiple-expert development. The glossary support system can be expected to emulate a hyper-expert notation scheme where the terms defined can be a mixture of text and graphics.

Implementation

The KASH system has a considerable level of potential for expediting the knowledge acquisition process and would be most beneficial if it were made widely accessible to the knowledge engineers. This goal requires that KASH be implemented on workstations or personal computers (PCs) because of their availability to the individual users without being subjected to the limited resources of larger systems. • A graphical interface will be necessary to provide a robust environment that will be both useful and meaningful to the end users. The

current state-of-the-art offers one such public domain technology, X-Windows. X-Windows is a portable, network transparent windowing system that is widely available on UNIX machines. An X-workstation or terminal (e.g., PC, Sun, MIPS, or Dec) would provide the industry standard look-and-feel interface, thus making KASH highly portable because the interface codes would not be required to be redeveloped between system architectures (see Figure 10). • The KASH system will be coded in the programming language C to achieve the primary goal of portability.

The initial target expert system shell for which KASH will generate rules will be CLIPS (C Language Production System). CLIPS was developed by the Mission Planning Group at NASA/JSC and is a very good candidate expert system shell. CLIPS provides high portability, low costs, and easy integration with existing conventional software systems. The availability of the CLIPS source code will prove beneficial because KASH will be able to treat the system *not* as a black box, but as an integrated component of the whole development environment. Furthermore, CLIPS as a C language callable library of functions, will provide a good testing platform for knowledge acquisition for embedded intelligent systems.

Conclusion

KASH is a general purpose knowledge acquisition shell that will acquire information about a domain from an expert. Since KASH has been designed to support both analysis and synthesis applications, it may be applied to a broad range of systems including planning, design, classification, scheduling, decision support, and diagnosis where complex knowledge is often acquired from multiple domain experts. The three modules defined in KASH will provide the capability to acquire knowledge in context (i.e., customized for each domain): *Concept Formulation* will structure and elicit the knowledge from a

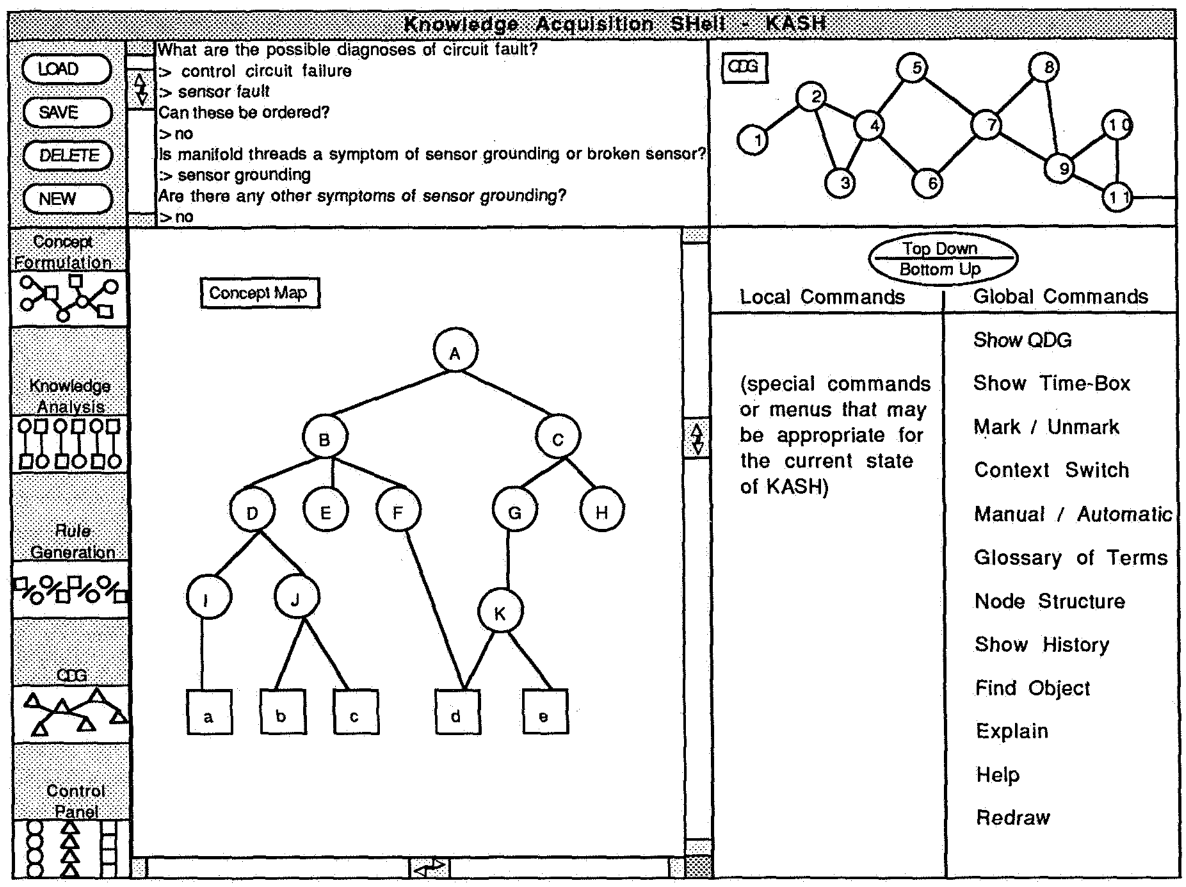

Figure 10. KASH User Screen

domain expert into a concept map; *Knowledge Analysis* will validate the concept map; and *Rule Generation* will produce a rule set for an expert system based on the concept map. Additionally, temporal knowledge acquisition will be accomplished through modifications to the concept maps using a time-box and time-line analysis to display the interval units. All the information elicited from a domain expert will be guided by a question dependency graph (QDG). The QDG will be developed by a knowledge engineer to separate the control knowledge from the application knowledge. Thus, the QDG will be reconfigured and customized across applications and domain experts. Furthermore, KASH will have the resources to produce a knowledge base that can be used to generate alternative rule sets for different expert system shells.

References

Allen, J.F. (1983). "Maintaining knowledge about temporal intervals," *Communications of the ACM*, Vol 26, No. 11. pp. 832-843.

Ausubel, D.P., J.D. Novak, and H. Hanesian (1978). *Educating Psychology: A Cognitive View*, 2nd edition, New York: Holt Rinehart, and Winston. Reprinted, 1986 New York: Warbel and Peck.

Cochran, E.L. (1988). "KLAMShell: A domain-specific knowledge acquisition shell," Technical Report CSDD-889-I6301-1, Honeywell, Golden Valley, MN.

Geesey, R.A. (1988). "A principled approach to tooling for temporal knowledge acquisition," BDM Technical Report, BDM/ROS-RG-06254-88. May 12, 1988.

Kahn, M.G., J.C. Ferguson, E.H. Shortliffe, and L.M. Fagan (1985). "Representation and Use of Temporal Information in ONCOCIN," *Proceedings Ninth Annual Symposium on Computer Applications in Medical Care*, IEEE Computer Society, pp. 172-176.

Ladkin, P. (1986). "Primitive and Units for Time Specification," *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, pp. 354-359.

LaFrance, M. (1987). "The knowledge acquisition grid: A method for training knowledge engineers," *Int. Journal of Man Machine Studies*, 26, 245-255.

Marcus, S. (1988). "SALT: A knowledge-acquisition tool for propose-and-revise systems," In Marcus, S. (ed.), *Automating Knowledge Acquisition for Expert Systems*. Kluwer Academic Publishers, Boston, pp. 81-122.

McGraw, K.L. and C.R. Westphal (1988). "Accommodating Multiple Expert Input During Knowledge Acquisition: Integrating Hypertext with a Knowledge Acquisition Tool," *Sixth Annual Intelligence Community AI Symposium*, Langley, VA.

Musen, M.A. (1989). "Knowledge acquisition at the metalevel: Creation of custom-tailored knowledge acquisition tools," *SIGART Newsletter*, No. 108, April. pp. 45-55.

Novak, J.D. (1989). "Helping students learn how to learn: A view from a teacher-researcher," *Proceedings of the Third Congress on Research and Teaching of Science and Mathematics*, Santiago de Compostela, Spain.

Novak, J.D. and B.D Gowin (1984). *Learning How to Learn*, New York: Cambridge University Press.

Rodi, L.L., J.A. Pierce, and R.E. Dalton (1989). "Putting the expert in charge: Graphical knowledge acquisition for fault diagnosis and repair," *SIGART Newsletter*, No. 108, April. pp. 56-62.

Simon H.A. (1969). *Sciences of the Artificial*, Cambridge, MA. MIT Press.

Westphal, C.R. and L.H. Reeker (1990). "Reasoning and representation mechanisms for multiple-expert knowledge acquisition," in *Knowledge Acquisition: Current Practices and Trends*, McGraw, K.L. and C.R. Westphal (eds.), Ellis Horwood, Chichester, England.

Westphal, C.R. and D. Tran (1990). "KASH: A general purpose knowledge acquisition shell," *Proceedings of the Florida Artificial Intelligence Symposium*, Cocoa Beach, FL.